AC

Requested Patent: EP0851353A1

Title: MEMORY MANAGEMENT IN A COMPUTER SYSTEM ;

Abstracted Patent: EP0851353 ;

Publication Date: 1998-07-01 ;

Inventor(s):

WELKER MARK W (US); BONOLA THOMAS J (US); MORIARTY MICHAEL P (US) ;

Applicant(s): COMPAQ COMPUTER CORP (US) ;

Application Number: EP19970310691 19971231 ;

Priority Number(s): US19960777781 19961231 ;

IPC Classification: G06F12/02 ; G09G1/16 ;

Equivalents: JP10247163

ABSTRACT:

A method and apparatus of allocating memory space in a main memory of a computer system to a unified memory architecture device. The main memory is associated with a physical address space. A required linear address range is determined for the video card, and the linear address range is mapped to scattered portions in the physical address space. A page table is created containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the device. The page frames are non-contiguous blocks of the main memory. The device is associated with a linear address space. The frame numbers are loaded into a translation look-aside buffer (TLB) for converting a linear address in the linear address space to a physical address in the physical address space.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 851 353 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.07.1998 Bulletin 1998/27

(51) Int Cl.[6]: **G06F 12/02**, G09G 1/16

(21) Application number: **97310691.7**

(22) Date of filing: **31.12.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **31.12.1996 US 777781**

(71) Applicant: **Compaq Computer Corporation**
Houston Texas 77070 (US)

(72) Inventors:
- **Welker, Mark W.**
  Spring, Texas 77389 (US)
- **Moriarty, Michael P.**
  Spring, Texas 77379 (US)
- **Bonola, Thomas J.**
  Tomball, Texas 77375 (US)

(74) Representative: **Brunner, Michael John et al**
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Memory management in a computer system

(57) A method and apparatus of allocating memory space in a main memory of a computer system to a unified memory architecture device. The main memory is associated with a physical address space. A required linear address range is determined for the video card, and the linear address range is mapped to scattered portions in the physical address space. A page table is created containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the device. The page frames are non-contiguous blocks of the main memory. The device is associated with a linear address space. The frame numbers are loaded into a translation look-aside buffer (TLB) for converting a linear address in the linear address space to a physical address in the physical address space.

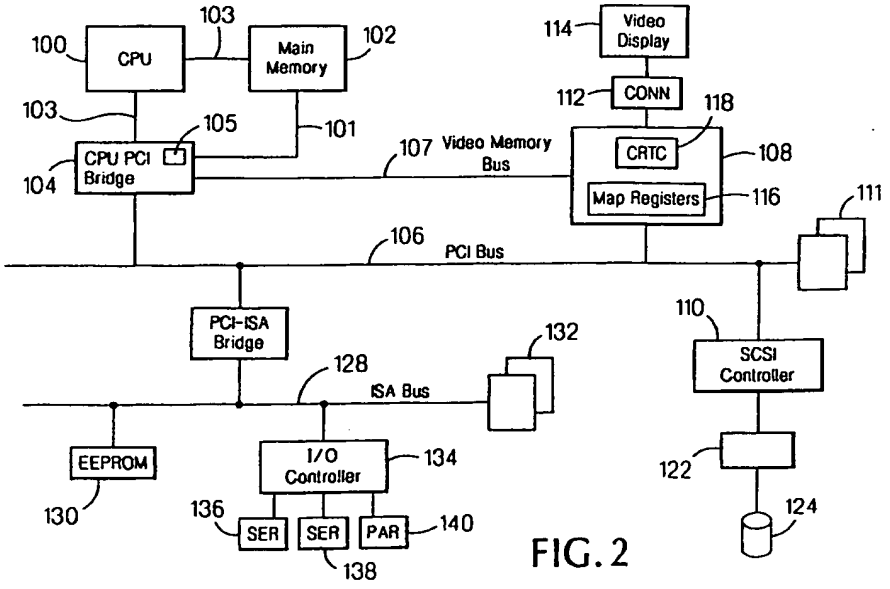

FIG. 2

EP 0 851 353 A1

Description

Background

The invention relates to memory management in a computer system.

A computer system includes a main memory (typically implemented with dynamic random access memories or DRAMs) used to store code and data information. Generally, a computer system includes several layers of buses, including a host bus, a Peripheral Component Interconnect (PCI) bus, and an expansion bus such as the Industry Standard Architecture (ISA) or Extended Industry Standard Architecture (EISA) bus. The main memory is used as a common resource accessible to system devices on the buses. Typically, the main memory is located on the host bus along with the central processing unit (CPU) for optimal performance of the computer system because the CPU requires fast access to the main memory.

Certain other system devices may also require relatively fast access to a memory device. One such system device is the video card, which typically includes a video memory (or frame buffer) for storing video data. Local storage of the video data allows a video controller in the video card to refresh or update a video display within the required amount of time to avoid flickering problems.

To avoid the need for a dedicated frame buffer on the video card, the VESA (Video Electronics Standards Association) Unified Memory Architecture (VUMA) was developed, which is described in "VESA Unified Memory Architecture (VUMA) Standard Hardware Specifications," Version 1.0 (March 8, 1996), and which is hereby incorporated by reference. VUMA is a computer system memory architecture in which the main memory is "physically" shared between the system (including the CPU and other devices) and a peripheral device (a "VUMA device"). Typically, a system device other than the CPU accesses the main memory through a direct memory access (DMA) controller over the ISA or EISA expansion bus, and a PCI bus device accesses the main memory in a memory transaction over the PCI bus. A VUMA device, on the other hand, is directly coupled to the main memory bus, as is the CPU, which improves memory access times for the VUMA device. One such VUMA device is a video card.

Referring to Fig. 1, a VUMA device 18 is connected to a Peripheral Component Interconnect (PCI) bus 12 in a computer system. The VUMA device 18 is also connected to a main memory 16 (having a storage capacity of, for example, 8 megabytes) over a memory bus 20, which is further connected to a core logic circuit 14 (including a bridge to the PCI bus 12 and a memory controller). The core logic circuit 14 acts as the interface b tween a CPU 10 and the PCI bus 12 and the main memory 16. Other devices (not shown) are also connected to the PCI bus 12.

The main memory 16 is shared between the system (including the CPU 10 and other PCI bus devices) and the VUMA device 18, which can directly acc ss the main memory 16 over the memory bus 20. A portion 22 of the main memory 16 is allocated as the memory for the VUMA device 18 ("the VUMA memory"), and the remainder of the main memory 16 is allocated to the rest of the computer system.

The VUMA memory 22 is mapped to the top of the main memory 16 as a single, contiguous region. Thus, for example, if the VUMA device 18 requires 2 megabytes (MB) of memory space, a 2-MB contiguous region of the main memory 16 is mapped to the VUMA memory 22, leaving the remaining 6-MB region for system use.

Summary

In general, in one aspect, the invention features a method of allocating memory space in a main memory of a computer system to a unified memory architecture device. The main memory is associated with a physical address space. A required linear address range is determined for the device, and the linear address range is mapped to scattered portions in the physical address space.

Implementations of the invention may include one or more of the following features. A page table is created containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the device. The page frames are in non-contiguous blocks of the main memory. The frame numbers are loaded into a translation look-aside buffer for converting a linear address in the linear address space to a physical address in the physical address space. The device includes a video card.

In general, in another aspect, the invention features a method of storing video data of a video card in a main memory of a computer system. Non-adjacent portions of the main memory are allocated, and the allocated portions are used to store the video data.

Implementations of the invention may include one or more of the following features. A page table is created containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the video card. The page frames are in non-contiguous blocks of the main memory. The video card is associated with a linear address space. The frame numbers are loaded into a translation look-aside buffer for converting a linear address in the linear address space to a physical address in the physical address space.

In general, in another aspect, the invention features a method of mapping a linear address of a unified memory architecture device to a physical address in a main memory in a computer system. Page fram numbers are stored in map registers, wherein the page frame numbers correspond to non-adjacent page frames in the main memory. One of the map registers is accessed with

th linear address to retrieve a portion of the physical address.

In general, in another aspect, the invention features a computer system including a main memory having a physical address space. A device is coupled to the main memory and allocated a linear address space. A controller is adapted to map the linear address space of the device to scattered portions in the physical address space.

Implementations of the invention may include one or more of the following features. A page table accessible by the controller contains page frame numbers representing page frames in the main memory allocated to the device. The page frames are in non-contiguous blocks of the main memory. The device includes a translation look-aside buffer. The controller is adapted to load the frame numbers of the page table into the translation look-aside buffer for converting a linear address to a physical address. The device includes a video card. The device is allocated to multiple regions in the main memory, the multiple regions being accessible by the device. Multiple page tables are accessible by the controller, with each page table containing page frame numbers representing page frames in the main memory corresponding to an allocated region.

Implementations of the invention may include one or more of the following advantages. Memory space in a main memory allocated to a computer system device can be scattered throughout the main memory without the need to allocate a contiguous region of the memory. Main memory space can be more effectively utilized by avoiding the requirement of allocating a contiguous portion of the main memory for any one particular device. By scattering the memory allocated to the device in multiple banks of the main memory, concurrent access to the different memory banks by the device and by other system devices is enabled.

Other advantages and features will be apparent from the following description and claims.

Description

Fig. 1 is a block diagram of a portion of a computer system.

Fig. 2 is a block diagram of a computer system in which the main memory is shared between a video card and the rest of the system.

Fig. 3 is a block diagram of a memory system in the computer system.

Fig. 4 is a block diagram of the memory addressing scheme in the computer system.

Fig. 5 is a flow diagram of main memory allocation.

Fig. 6 is a diagram of a memory descriptor list.

Fig. 7 is a block diagram of portions of a video card.

Fig. 8 is a block diagram of read-ahead logic in the video card.

Fig. 9 is a flow diagram of main memory allocation in which multiple frame buffers are to be allocated in main memory for the video card.

Referring to Fig. 2, a computer system includes a central proc ssing unit (CPU) 100, such as the Pentium Pro processor, and a main memory 102. The main memory 102 can be implemented with a high bandwidth multichannel memory architecture, such as Rambus dynamic random access memories (RDRAMs) arranged in multiple RDRAM channels 1-3. The main memory 102 can have a combined total storage capacity of 16 megabytes (MB), for example. Each RDRAM channel is accessible by the CPU 100, a video card 108, or a device on a Peripheral Component Interconnect (PCI) bus 106.

The computer system implements the VESA Unified Memory Architecture (VUMA) in which the video card 108 is the VUMA device. Access to the main memory 102 is controlled by a memory controller 105 in a CPU-PCI bridge 104 connected to the PCI bus 106. Requests for the main memory 102 to the memory controller 105 can come over a host bus 103 (from the CPU 100), a video memory bus 107 (from the video card 108), or the PCI bus 106 (from PCI bus masters).

Referring to Fig. 3, the memory system includes the memory controller 105 connected to three RDRAMs 102A, 102B, and 102C over RDRAM buses 101A, 101B, and 101C, respectively. Each RDRAM channel is associated with a unique address range. The memory controller 105 includes three memory interface control (MIC) blocks 142A, 142B, and 142C corresponding to the three RDRAM buses 101A, 101B, and 101C, respectively. Each MIC block 142 includes its own arbiter, page-hit detection and control logic, and refresh counter and timer. Using multiple RDRAM channels and MIC blocks 142, concurrent access to the main memory 102 by more than one device is allowed. Thus, for example, the CPU 100 can be accessing an address location over one RDRAM bus while the video card 108 can be concurrently accessing another address location over another RDRAM bus, so long as the two address locations are located in two different RDRAM ranges.

Referring again to Fig. 2, as a VUMA device, the video card 108 performs video memory transactions over the video memory bus 107. The video card 108 is coupled to a video display 114 through a connector 112 and includes a CRT controller 118 and map registers 116. The map registers 116 map the linear memory space allocated to the video card 108 to the physical memory space of the main memory 102. A linear address generated by the CRT controller 118 selects one of the map registers 116 to retrieve a physical memory address presented over the video memory bus 107 to perform a main memory access.

A portion (e.g., 1 MB, 2 MB, or 4 MB) of the main memory 102 (hereinafter referred to as "video memory") is allocated to the video card 108. To provide greater flexibility, the video memory is scattered throughout the main memory 102. Thus, the video memory is mixed with the system memory, thereby avoiding the need to allocate a contiguous region of space in main memory

to the video memory. In addition, the ability to scatter the memory space of an UMA device throughout the main memory 102 also improves ov rall system performance, since a more dynamic memory allocation scheme can be implemented.

Other components of the computer system include a SCSI controller 110 (or alternatively, an IDE controller) connected to the PCI bus 106, the SCSI controller 110 being coupled to a mass storage system 124 through a SCSI connector 122. The mass storage system 124 can be a hard disk drive array or a CD-ROM drive. PCI slots 111 also exist on the PCI bus 106 for connection to PCI peripheral cards.

The computer system also includes an Industry Standard Architecture (ISA) expansion bus 128. PCI bus devices communicate with ISA bus devices through a PCI-ISA bridge 126. A non-volatile memory 130 (such as an electrically erasable programmable read-only memory or EEPROM) is connected to the ISA bus 128, as are ISA slots 132 for receiving ISA expansion devices (not shown) . In addition, an input/output (I/O) controller 134 is connected to the ISA bus 128. The I/O controller 134 provides interfaces to two serial ports 136 and 138 (which can be connected to a mouse and a keyboard) and a parallel port 140 (which can be connected to a printer).

Software running on the computer system includes applications programs and an operating system, such as the Windows NT operating system. During computer system initialization, basic input/output system (BIOS) code stored in the EEPROM 130 is executed to configure system components, including allocating memory address space, input/output (I/O) address space, and interrupt lines to the devices. Video BIOS code is also stored in the EEPROM 130 (or alternatively, in non-volatile memory in the video card) and is executed to configure the video card 108, including the CRT controller 118 and map registers 116.

Referring to Fig. 4, the video memory allocated to the video card 108 is mapped to non-adjacent physical page frames in the main memory 102. In the example shown in Fig. 4, the video memory is stored in page frames 300, 302, and 304 in the main memory 102, with the video memory page frames being scattered between page frames 306, 308, and 310 used by the rest of the computer system, including the CPU 100. The only requirement of the memory allocation scheme is that the memory allocated to the video card 108 be used only as video memory. Once physical page frames are allocated to the video memory, those physical page frames are reserved for use as video memory until the memory management process of the operating system changes the physical page allocation.

Additionally, the video memory is distributed throughout RDRAM channels 1, 2, and 3 of the main memory 102. By so distributing the video memory, system performance can further be enhanced by allowing concurrent access to the main memory 102 by the video card 108 over one RDRAM channel while another system device is accessing the main memory over another RDRAM channel.

The video memory space is represented as a linear logical space addressable by the CRT controller 118. The size of the linear logical space can be as large as any conventional video frame buffer (e.g., 1 MB, 2 MB, or 4 MB) or as small as a display line, depending on the requirements of the video system. The map registers 116 in the video card 108 form a translation look-aside buffer (TLB) 143 that maps the linear logical space of the frame buffer to physical page frames in the main memory 102. On the CPU side, a TLB 144 maps the linear logical space seen by the CPU 100 to physical memory space.

Referring to Fig. 5, the linear logical space of the video frame buffer is first allocated at step 202 during initialization of the video device driver in the computer boot-up sequence. The memory allocation is performed by a memory management process in the operating system using standard memory management features, which are specific to the operating system used in the computer, such as Windows NT. The video card 108 includes information storage circuitry accessible by the video device driver to determine the amount of linear address range required by the video card 108. Such information storage circuitry includes non-volatile memory or an application-specific integrated circuit (ASIC) for storing such information as the number of physical memory pages required by the video card 108. Alternatively, the video device driver can retrieve the version ID of the video card 108, and based on the retrieved version ID, determine the required linear address range.

For example, if the video display 114 has dimensions of 1024 x 768, with 16 bits per pixel, then approximately 1.57 megabytes (MB) of video memory space is required. The actual amount of memory space allocated is a multiple of the page frame size (which is 4 kilobytes or kB in the Windows NT operating system). Once the video device driver determines the linear address range required for the video card 108, it transmits a request to the memory management process of the operating system to allocate the required amount of memory. The request also indicates that the allocation is to be page aligned, i.e., 4-kB aligned. In response, the memory management process returns the address information related to the allocated page frames.

Next, the video device driver of the operating system creates at step 204 page tables or lists for mapping video linear address space to actual physical page frames in the main memory 102. If the operating system is the Windows NT operating system, for example, then a memory descriptor list (MDL), shown in Fig. 6, is created at step 204 for the video memory space. An MDL is created for each specific task under Windows NT. Tasks include active application programs. The MDL includes header information followed by a series of physical frame numbers, which correspond to available page

frames in the main memory 102. For each particular task, the page frame numbers contained in the MDL represent the page frames in the main memory 102 allocated to the task. The header information includes the starting linear address of the Windows task (e.g., the video memory) and the length of memory allocated to the task. For example, given a 1024 x 768 x 16 video display and a memory page size of 4 kB, the length specified in the MDL would be 1.57 MB.

After the MDL for the video memory is created at step 204, the physical frame numbers are written by the video display driver of the operating system at step 206 to the map registers 116 in the video card 108. Once the map registers 116 are programmed, then any linear address generated by the CRT controller 118 will be properly mapped to a physical memory address using the contents of the map registers 116.

Referring to Fig. 7, the CRT controller 118 generates linear address signals VAD[21:0]. The 10 most significant address bits, VAD[21:12], are used to select one of 1024 map registers 116 each storing a physical page frame number. Alternatively, if more map registers are required, then the CRT controller 118 can generate additional, more significant address bits. The physical page frame numbers stored in the map registers 116 provide the 20 most significant bits of the physical memory address PAD[31:12], i.e., the starting address of a physical page frame. The remaining 12 bits of the linear address, VAD[11:0], are connected directly to the 12 least significant bits of the physical memory address PAD[11:0] to provide the offset into the selected page frame.

Use of the map registers 116 to perform linear-to-physical address mapping provides flexibility in how the video memory space is to be allocated in the physical memory space. The operating system can scatter video memory over different page frames in the main memory 102 if a contiguous region of the main memory 102 is not available for the video memory. Without this ability to scatter video memory, the operating system could have to perform a complete re-allocation of the main memory 102 to create a contiguous region in the main memory for video when the display mode is initialized.

In certain video applications, such as high-end video animation, more than one frame buffer is used by a video system. Thus, for example, if two frame buffers are used in a video animation application, the video data stored in one frame buffer is being displayed while the video animation software running in the computer system writes video data to the other frame buffer. Thus, the computer system can alternately display the contents of the two frame buffers (while the other frame buffer is being updated) to create a superior animation effect on the video display.

Referring to Fig. 9, the two (or mor if necessary) frame buffers can be allocated in the main memory 102 using two or more separate MDLs in the video card 108, with one MDL defining the allocated physical page frames for each video frame buffer. Thus, for a multiple-frame buffer video system, th linear address space for each frame buffer is separately allocated at step 502 for each frame buffer, i.e., the total space to be allocated each frame buffer is determined. Next, the separate MDLs are created at step 504.

Next, at step 506, it is determined if one or multiple sets of map registers are used. If only one set of map registers in the video card 108 is used, then the contents of the map registers are alternately filled with the contents of the active one of the multiple MDLs by the animation application software during the animation process. In this scheme, contents of a first MDL is initially written at step 508 to the map register.

If multiple sets of map registers are used in the video card 108, with each map register dedicated to storing the frame numbers of a particular MDL, then the frame numbers of all the MDLs are written at step 510 to corresponding sets of map registers.

Each frame buffer is allocated to distinct page frames in the main memory 102. The frame buffers are scattered throughout the main memory 102 according to memory allocations performed by the memory management process of the operating system.

If multiple sets of map registers are used in the video card 108, then the CRT controller 118 provides a pointer to point to the appropriate set of map registers, i.e., the set corresponding to the frame buffer whose contents are currently being displayed. The pointer is updated by software to point to the frame buffer to be displayed.

If only one set of map registers is used, then the contents of the map registers must be updated with the frame numbers from the MDL of the next frame buffer to be displayed during the video display vertical retrace period. In operation, if two MDLs are used, for example, then the contents of one MDL (as determined by a pointer provided by the video device driver) are first written to the map registers for displaying the contents of the corresponding video display. When the second frame buffer is ready to be displayed, and during the vertical retrace period of the CRT controller 118, the video device driver points to the other MDL and loads its contents into the map registers for displaying the contents of the second frame buffer. This alternating scheme continues throughout the animation process.

A further feature of the video card 108 is its ability to fetch ahead for video data to improve video performance and efficiency. Generally, video data is processed as a sequential stream of data for display on the computer monitor 114, making the fetch-ahead operation very predictable. Referring to Fig. 8, the video card 108 includes read-ahead logic 169 (which includes a read-ahead controller 170 and a read-ahead counter 172) for opening the next page of video data from the main memory 102 while the CRT controller 118 is working on the current page of video data.

The read-ahead counter 172 receives and stores

the first linear address VAD[21:12] from the CRT controller 118 in response to assertion of a signal LOAD_COUNTER by the read-ahead controller 170. The read-ahead controller 170 provides a signal INC_COUNTER to increment the counter 172 (after the initial loading) and a signal RESET_COUNTER to clear the counter 172.

The read-ahead controller 170 controls the states of the signals LOAD_COUNTER and INC_COUNTER based on a signal PAGE_PROCESSED and a signal FIRST_PAGE_READ from the CRT controller 118. Upon assertion of the signal FIRST_PAGE_READ, which indicates that the CRT controller 118 is beginning a new, non-consecutive page read cycle, the read-ahead controller 170 asserts the signal LOAD_COUNTER to load in the value of the address VAD[21:12]. Thereafter, each assertion of the signal PAGE_PROCESSED, asserted by the CRT controller 118 when the CRT controller 118 has reached a "high water mark," e.g., it has processed half or some other predetermined portion of the currently retrieved video data, the read-ahead controller 170 asserts the signal INC_COUNTER to increment the counter 172. In addition, in response to assertion of the signal PAGE_PROCESSED, the read-ahead controller asserts a signal READ_AHEAD to a video memory bus interface block 176, which generates a hidden page read cycle on the video memory bus 107.

The address for a video memory bus transaction is provided by multiplexers 178 and 180. The multiplexer 178 selects between the linear address VAD[21:12] from the CRT controller 118 (a signal READ_AHEAD_SEL is low) and the address CNT_VAD [21:12] from the counter 172 (READ_AHEAD_SEL is high) for output as address MUX_VAD[21:12]. In similar fashion, the multiplexer selects between address signals VAD[11:0] from the CRT controller 118 and twelve "0s" for output as physical address PAD[11:0]. The address signals MUX_VAD[21:12] select one of the map registers 116 to output the physical address PAD[31:12]. The physical address PAD[31:0] is provided as the address in a video memory bus transaction. In a hidden page read cycle (generated for reading ahead), the physical address is {PAD[31:12, 000000000000}.

A hidden page read request on the video memory bus 107 is received by the memory controller 105, which in turn generates a hidden page read cycle to an appropriate one of the RDRAM channels 101. Such a hidden page read cycle does not retrieve data from the main memory 102, but merely serves to keep a page "open" in the memory controller 105.

Each MIC block 142A, 142B, or 142C includes a set of page address registers for storing a row address of the corresponding RDRAM channel 101A, 101B, or 101C. The RDRAMs making up the main memory 102 are addressed using the standard multiplex d row/column addressing scheme for DRAMs, in which a row address selects a page in a DRAM and a column address selects a bit in that page. In DRAM page mode, the row address is maintained constant while only th column addresses are changed, which reduces DRAM access time as row addr ss precharge time can b avoided. Keeping a page open involves storing a row address in on of the page address registers. Thus, if the next memory request is to the same page (i.e., has the same row address), then the page-hit detection and control logic of the MIC block 142 produces a hit indication, and a page mode cycle is generated on the corresponding RDRAM channel 101 to retrieve the requested data.

By keeping the video frame buffer page open in the memory controller 105, the next memory read cycle from the CRT controller 118 to the same page will allow the memory controller to quickly retrieve the page of data using the DRAM's page mode. The read-ahead feature reduces significantly memory access latency for the video card 108.

Other embodiments are within the scope of the following claims. For example, the memory mapping scheme can be extended to other types of peripheral devices which require memory.

Claims

1. A method of allocating memory space in a main memory of a computer system to a unified memory architecture device, wherein the main memory is associated with a physical address space, the method comprising:

   determining a required linear address range for the device; and
   mapping the linear address range to scattered portions in the physical address space.

2. The method of claim 1, further comprising:
   creating a page table containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the device.

3. The method of claim 2, wherein the page frames are in non-contiguous blocks of the main memory.

4. The method of claim 2, wherein the device is associated with a linear address space, the method further comprising:
   loading the frame numbers into a translation look-aside buffer for converting a linear address in the linear address space to a physical address in the physical address space.

5. The method of claim 1, wherein the device includes a video card.

6. A method of storing video data of a video card in a main memory of a computer system, comprising:

allocating non-adjacent portions of the main memory; and
using the allocated portions to store the video data.

7. The method of claim 6, further comprising:
creating a page table containing page frame numbers corresponding to page frames in the main memory, the page frames being allocated to the video card.

8. The method of claim 7, wherein the page frames are in non-contiguous blocks of the main memory.

9. The method of claim 7, wherein the video card is associated with a linear address space, the method further comprising:
loading the frame numbers into a translation look-aside table for converting a linear address in the linear address space to a physical address in the physical address space.

10. A method of mapping a linear address of a unified memory architecture device to a physical address in a main memory in a computer system, the method comprising:

storing page frame numbers in map registers, wherein the page frame numbers correspond to non-adjacent page frames in the main memory; and
accessing one of the map registers with the linear address to retrieve a portion of the physical address.

11. A computer system, comprising:

a main memory having a physical address space;
a device coupled to the main memory and allocated a linear address space;
a controller adapted to map the linear address space of the device to scattered portions in the physical address space.

12. The computer system of claim 11, further comprising:
a page table accessible by the controller, the page table containing page frame numbers representing page frames in the main memory allocated to the device, wherein the page frames are in non-contiguous blocks of the main memory.

13. The computer system of claim 12, wherein the device includes a translation look-aside buffer, the controller being adapted to load the frame numbers of the page table into the translation look-aside buffer for converting a linear address to a physical address.

14. The computer system of claim 11, wherein the device includes a video card.

15. The computer system of claim 11, wherein the device is allocated to multiple regions in the main memory, the multiple regions being accessible by the device, the computer system further comprising:
multiple page tables accessible by the controller, each page table containing page frame numbers representing page frames in the main memory corresponding to an allocated region.

16. The computer system of claim 15, wherein the device includes a video card, and wherein each allocated region in the main memory corresponds to a frame buffer for the video card.

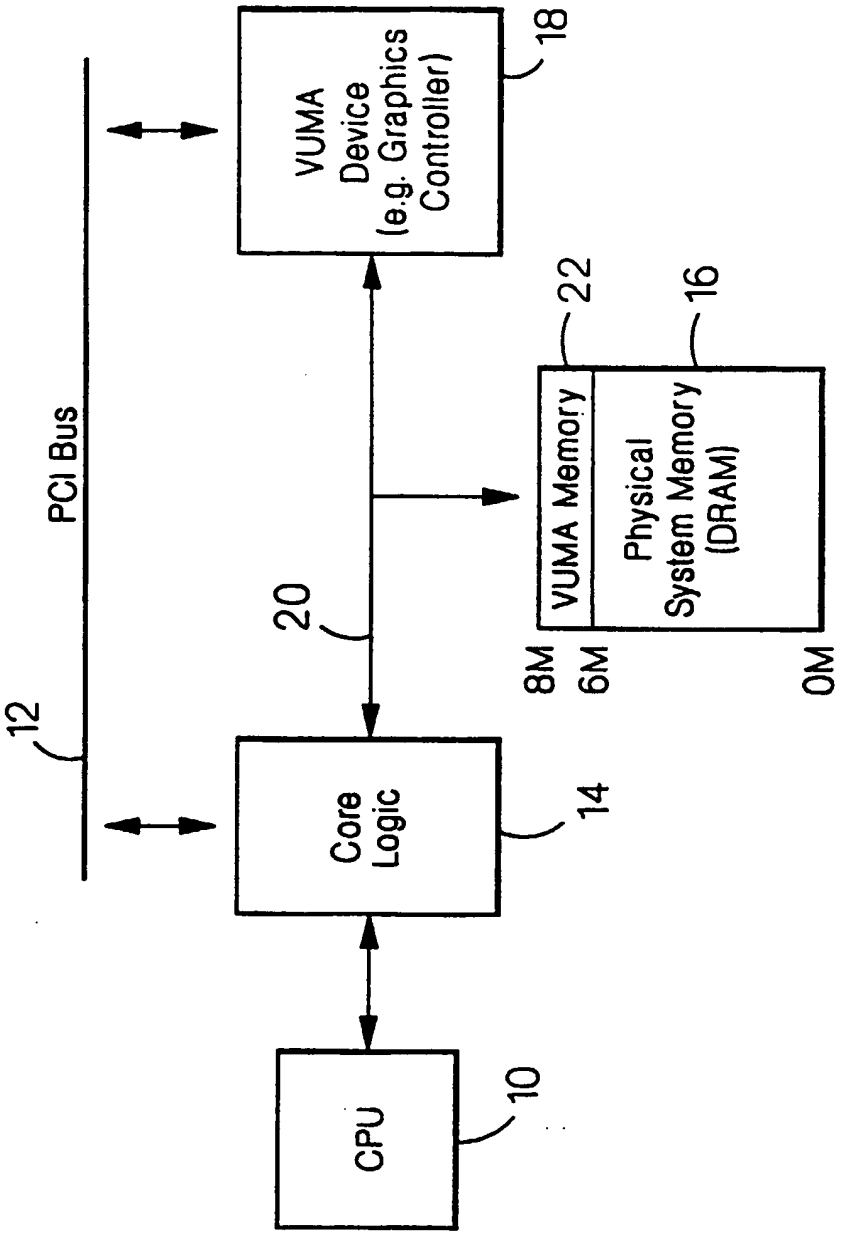

FIG. 1
(PRIOR ART)

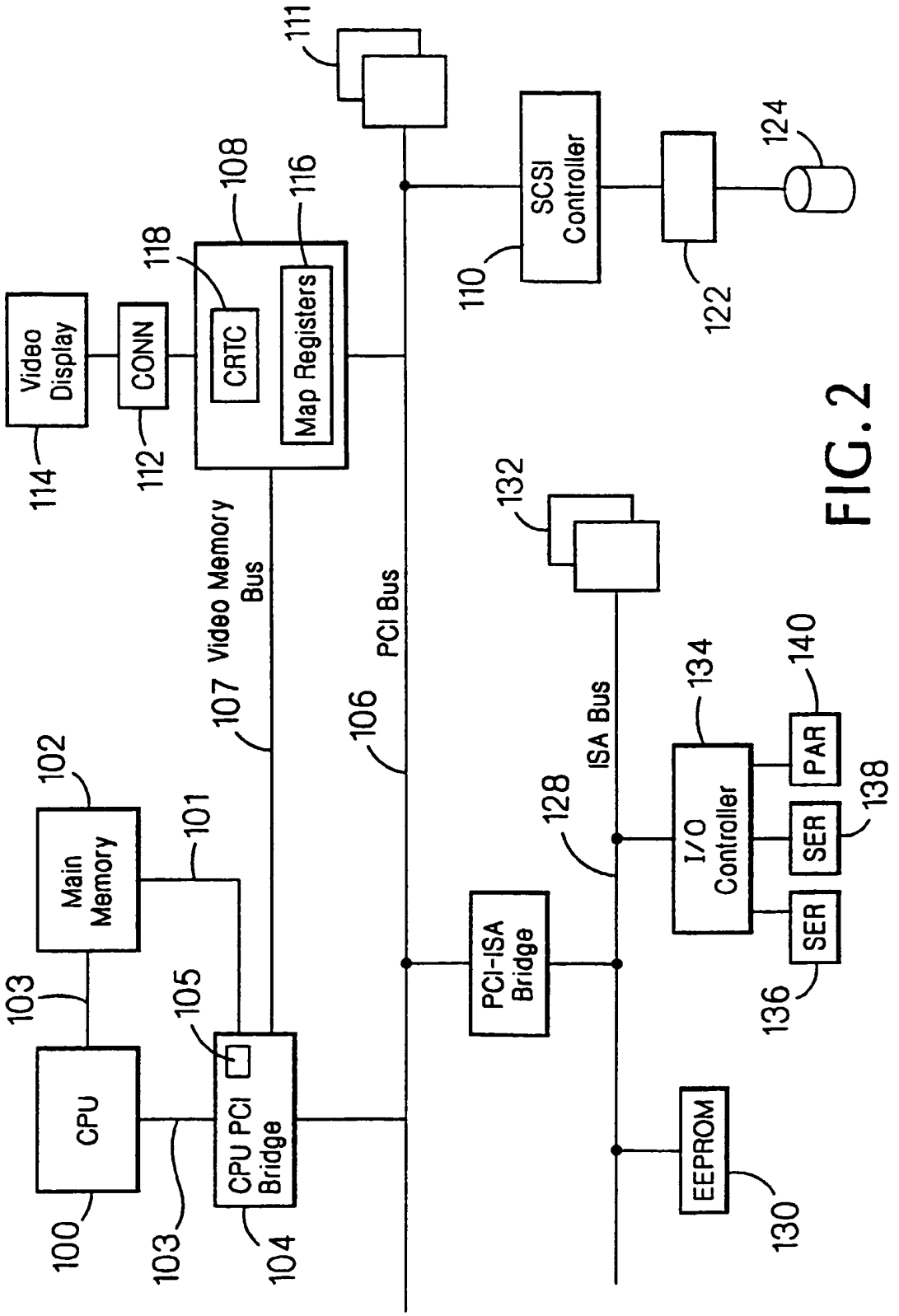
CPU
Main Memory
CPU PCI Bridge
PCI-ISA Bridge
I/O Controller
SCSI Controller
EEPROM
PAR
SER
SER
CRTC
CONN
Video Display
Map Registers
PCI Bus
ISA Bus
Video Memory Bus
100
101
102
103
103
104
105
106
107
108
110
111
112
114
116
118
122
124
128
130
132
134
136
138
140

FIG. 2

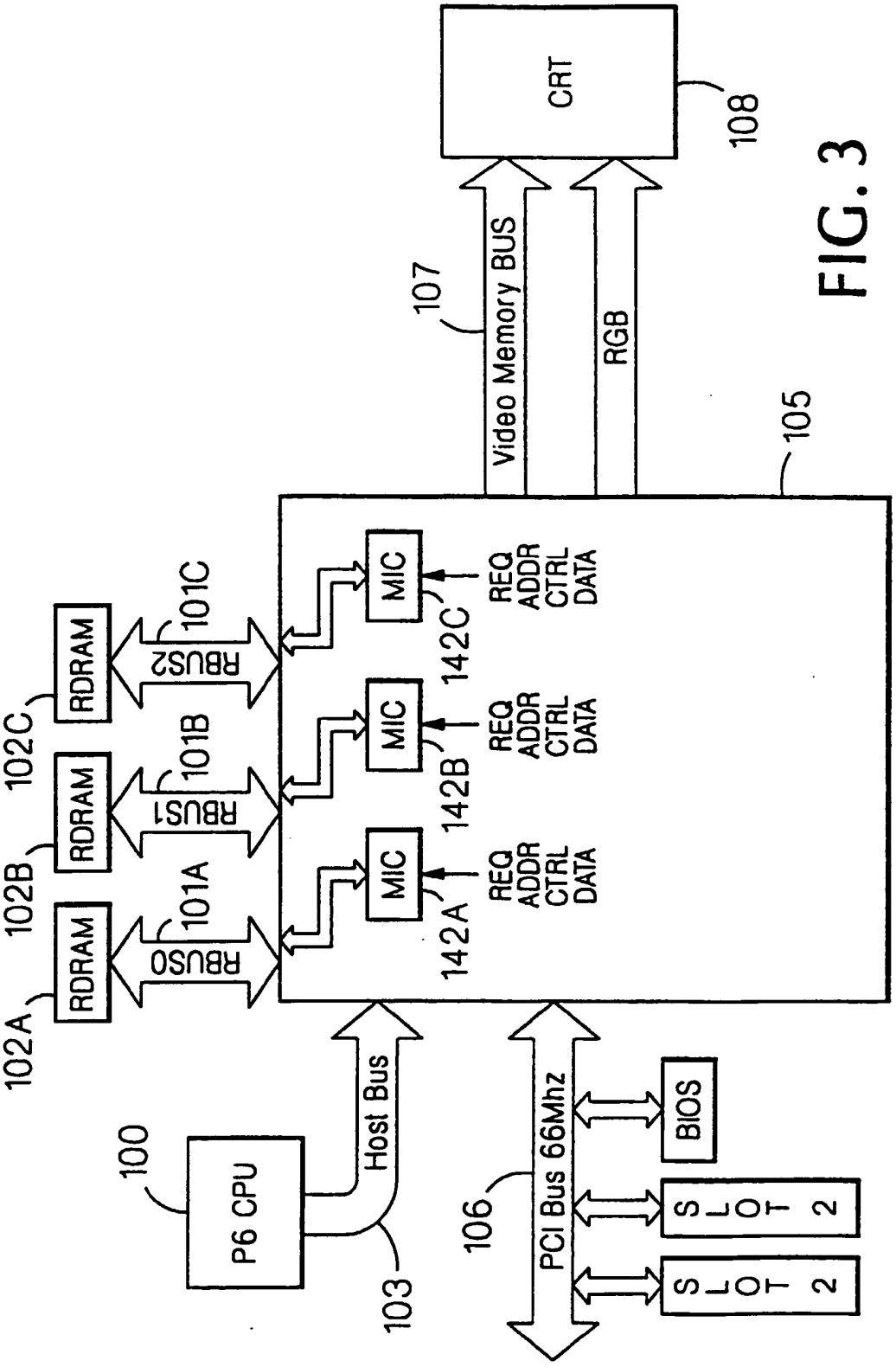
CRT
108
107
Video Memory BUS
RGB
105
RDRAM
102A
RDRAM
102B
RDRAM
102C
RBUS0
101A
RBUS1
101B
RBUS2
101C
MIC
142A
MIC
142B
MIC
142C
REQ
ADDR
CTRL
DATA
P6 CPU
100
Host Bus
103
PCI Bus 66Mhz
106
BIOS
SLOT 2
SLOT 2

FIG. 3

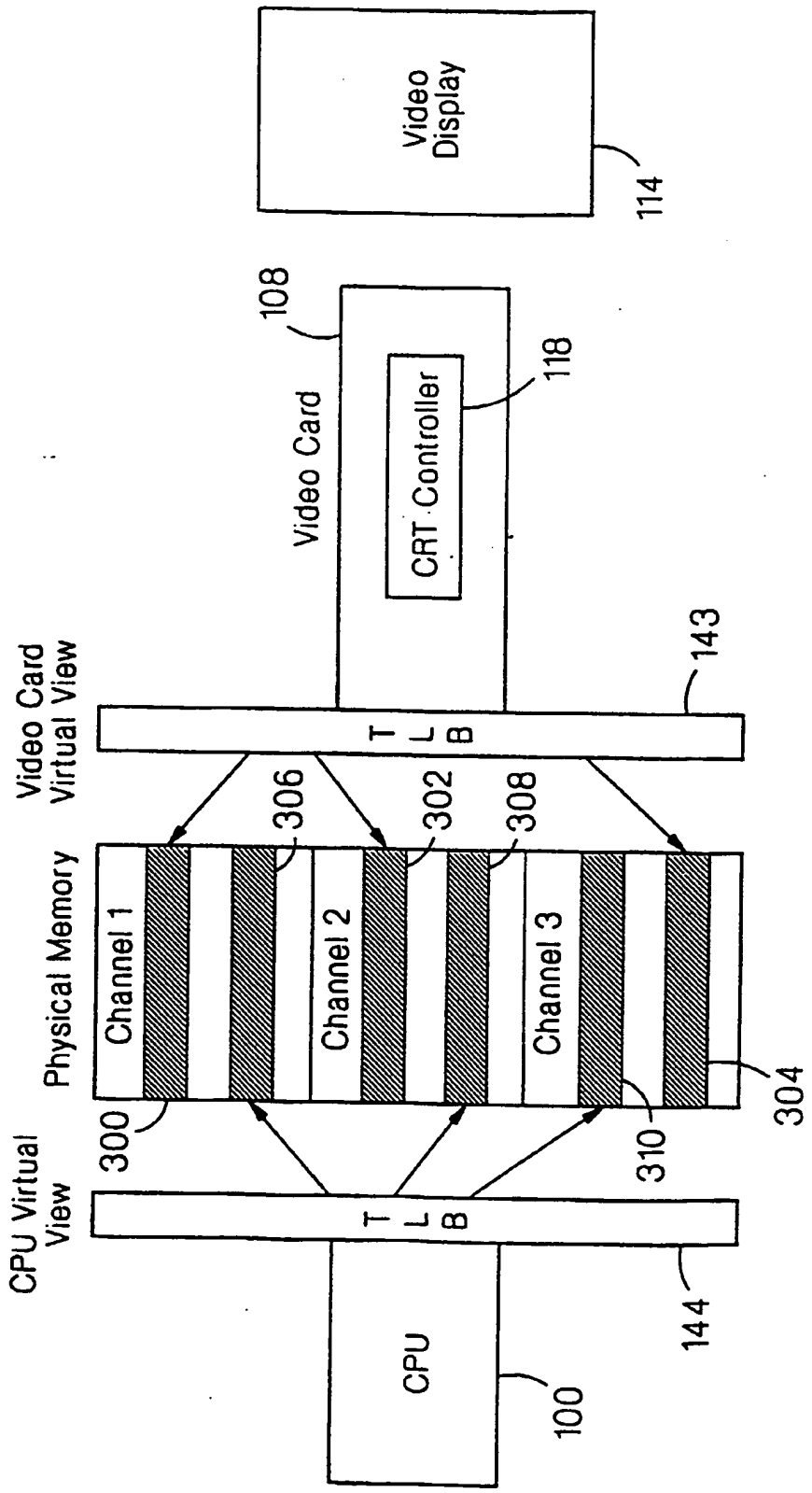
Video Display
114
108
Video Card
CRT Controller
118
Video Card Virtual View
143
T L B
306
302
308
Physical Memory
Channel 1
Channel 2
Channel 3
300
304
310
CPU Virtual View
T L B
144
CPU
100

FIG. 4

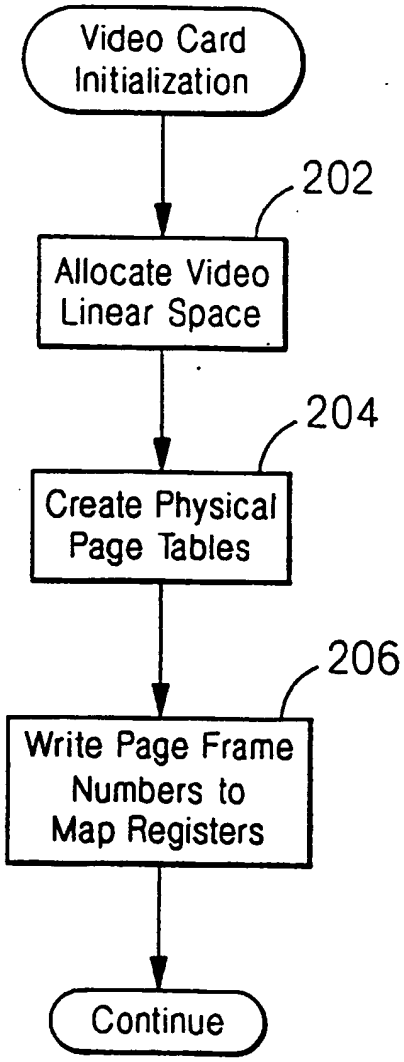
Video Card Initialization
202
Allocate Video Linear Space
204
Create Physical Page Tables
206
Write Page Frame Numbers to Map Registers
Continue

FIG. 5

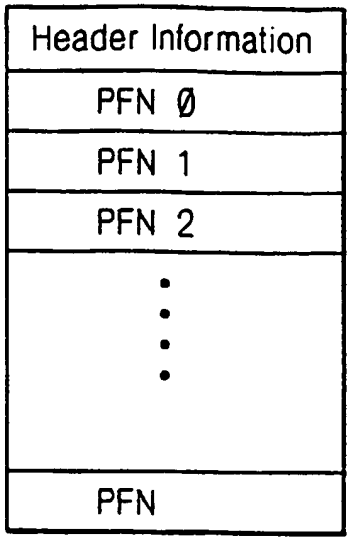
Header Information
PFN Ø
PFN 1
PFN 2
PFN

FIG. 6

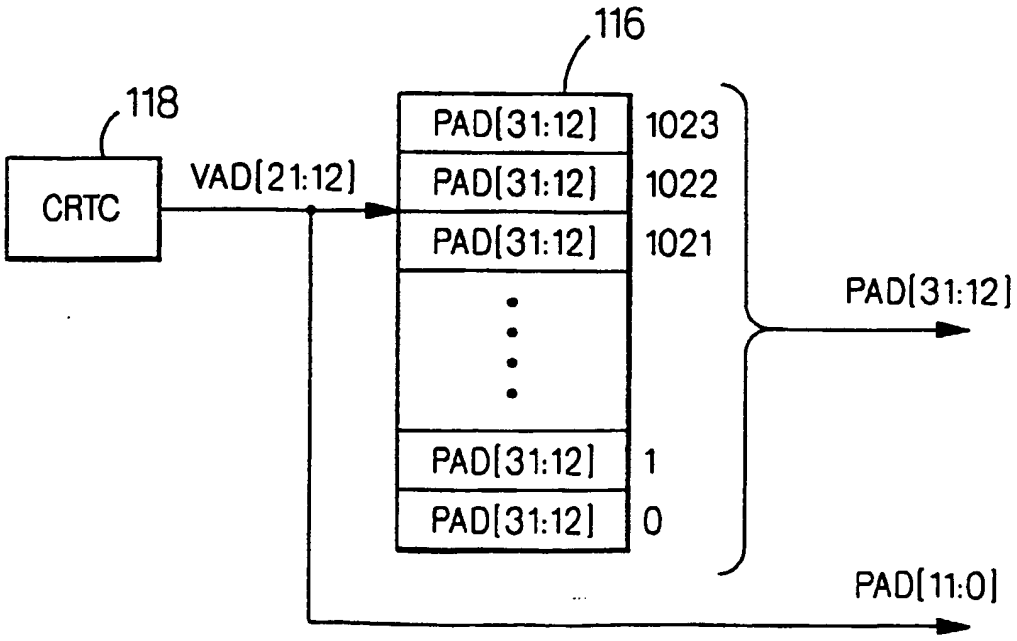
116
118
CRTC
VAD[21:12]
PAD[31:12]
PAD[31:12]
PAD[31:12]
PAD[31:12]
PAD[31:12]
PAD[31:12]
1023
1022
1021
1
0
PAD[31:12]
PAD[11:0]

FIG. 7

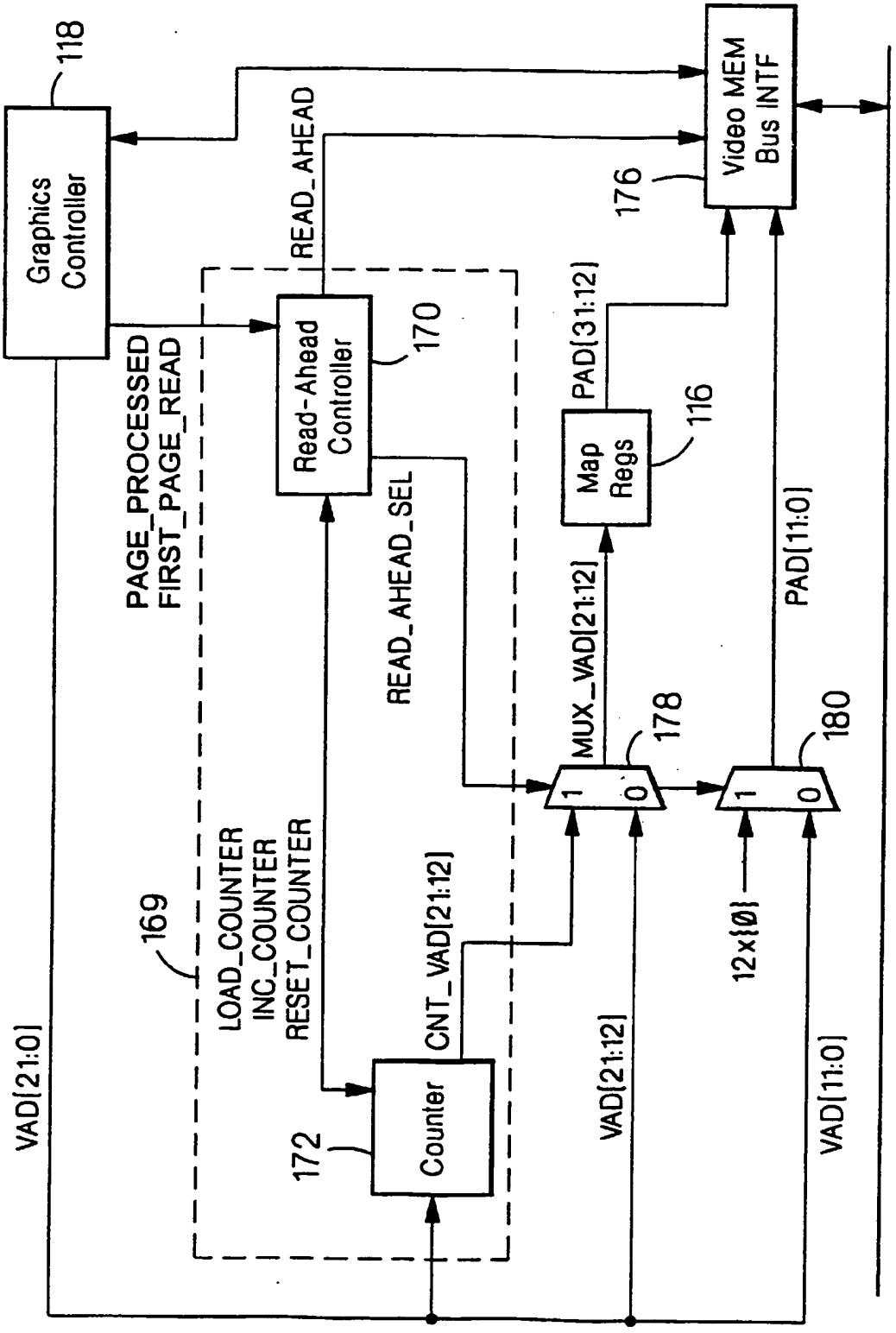
118
Graphics Controller
Video MEM Bus INTF
176
READ_AHEAD
PAGE_PROCESSED
FIRST_PAGE_READ
Read-Ahead Controller
170
PAD[31:12]
Map Regs
116
READ_AHEAD_SEL
MUX_VAD[21:12]
178
180
PAD[11:0]
169
LOAD_COUNTER
INC_COUNTER
RESET_COUNTER
CNT_VAD[21:12]
VAD[21:0]
172
Counter
VAD[21:12]
12x{0}
VAD[11:0]

FIG. 8

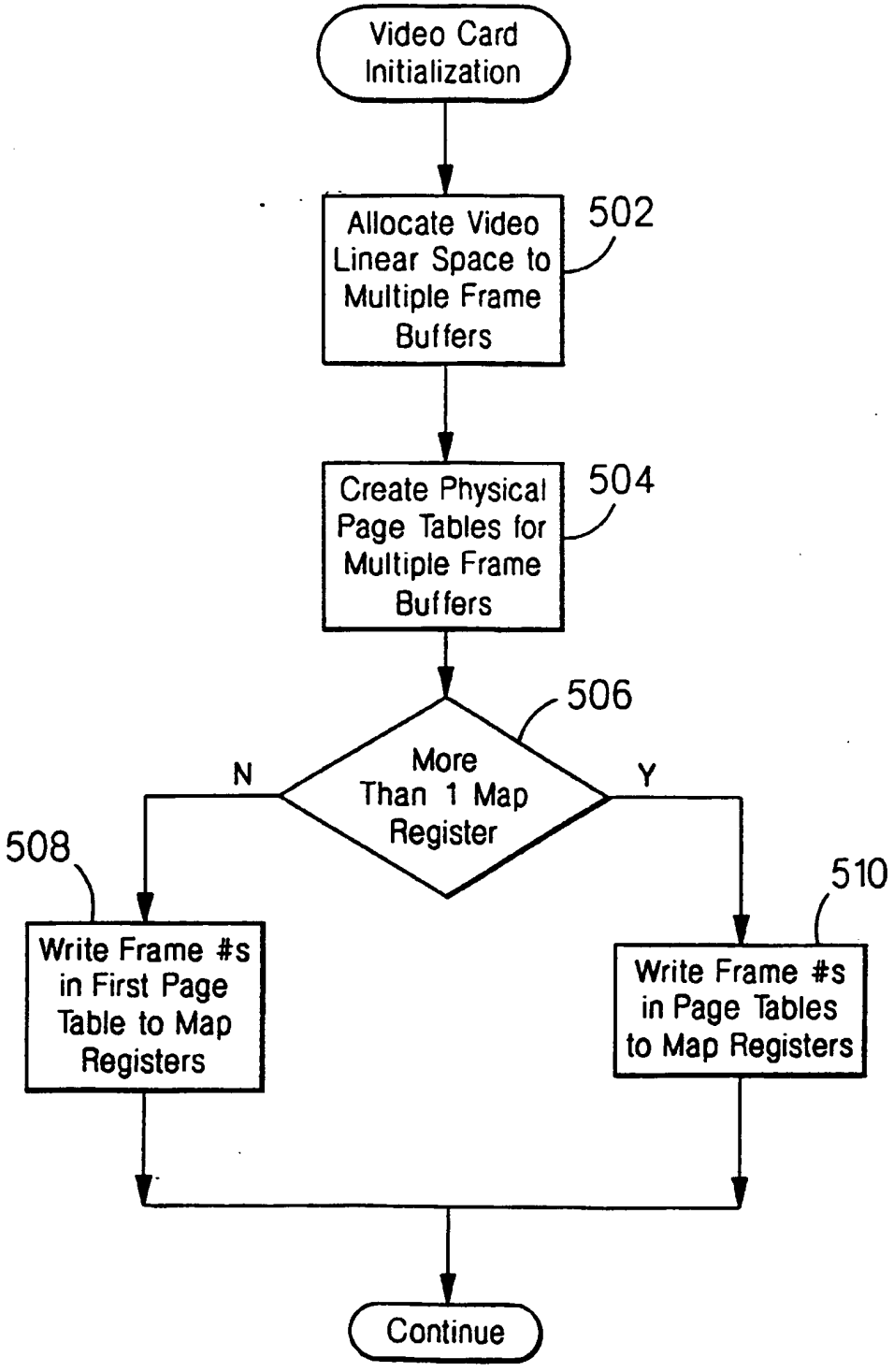
Video Card Initialization
Allocate Video Linear Space to Multiple Frame Buffers
502
Create Physical Page Tables for Multiple Frame Buffers
504
More Than 1 Map Register
506
N
Y
508
Write Frame #s in First Page Table to Map Registers
510
Write Frame #s in Page Tables to Map Registers
Continue

FIG. 9

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 31 0691

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP 0 338 416 A (IBM) * column 4, line 49 - column 6, line 1 * * column 7, line 42 - column 8, line 24; figure 1 *	1-16	G06F12/02 G09G1/16
E	EP 0 829 820 A (SILICON GRAPHICS INC) 18 March 1998 * the whole document *	1-16	
X	US 5 386 524 A (LARY RICHARD ET AL) * column 3, line 46 - column 4, line 42 *	1-4, 10-13,15	
A	US 5 454 107 A (LEHMAN JUDSON A ET AL) * abstract; figures 3,5 *	1-16	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F G09G

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	21 April 1998	Nielsen, O

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)